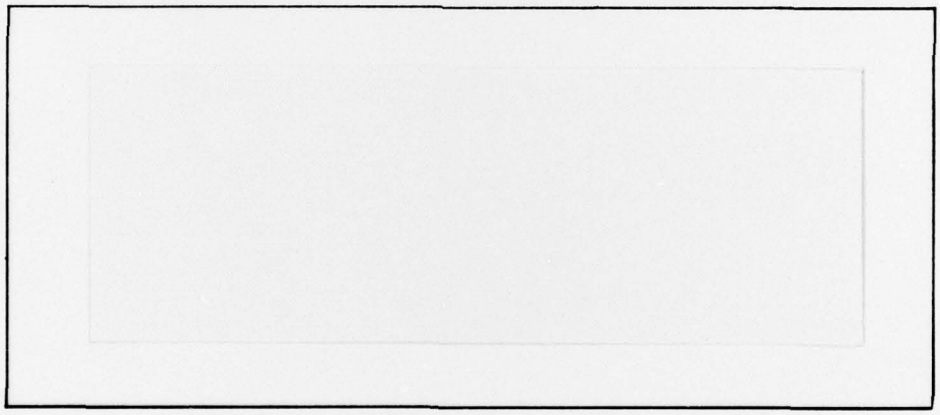
AD-A039 473 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 12/2
A LINEAR TIME ALGORITHM FOR DECIDING SECURITY.(U)
OCT 76 A K JONES, R J LIPTON, L SNYDER N00014-75-C-0752
UNCLASSIFIED RR-103 NL

| OF |
AD
A039473

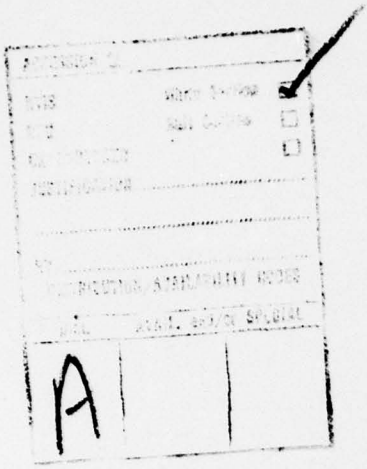
END

DATE
FILMED
6-77

12



YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE



A Linear Time Algorithm for Deciding Security

A. K. Jones,[†] R. J. Lipton[††] and L. Snyder[††]

Research Report #103

[†] Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213. This work was supported in part by NSF under grant DCR75-07251.

[††] Department of Computer Science, Yale University, New Haven, Connecticut 06520. This work was supported in part by the Office of Naval Research grant N00014-75-C-0752 and in part by NSF grant DCR74-24193.

A LINEAR TIME ALGORITHM FOR DECIDING SECURITY

A. K. Jones†
Carnegie-Mellon University
Pittsburgh, PA 15213
R. J. Lipton††
L. Snyder††
Yale University
New Haven, CT 06520

The Folklore is replete with stories of "secure" protection systems being compromised in a matter of hours. This is quite astounding since one is not likely to claim that a system is secure without some sort of proof to support the claim. In practice, proof is not provided and one reason for this is clear: although the protection primitives are apparently quite simple, they may potentially interact in extremely complex ways. Vague and informal arguments, therefore, often overlook subtleties that an adversary can exploit. Precision is not merely desirable for protection systems, it is mandatory.

Accordingly, this paper is devoted to the analysis of a specific protection system of both theoretical interest and practical interest. Theoretically, these problems are graph theoretic in flavor and they can be reasonably be viewed as generalizations of "transitive closure". Roughly these protection questions can be modeled as:

Given: A directed labeled graph G and a set of rewriting rules R.

Determine: Whether or not there is a sequence of graphs $G_1, G_2, \ldots, G_n$ such that $G = G_1$, G_n has property X, and G_{i+1} follows from G_i by some rule in R.

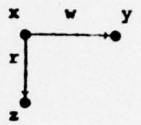
Here the G_i represent the protection state and property X encodes that there is a protection violation in G_n. Our goal then is to show that it is impossible to reach such a G_n, i.e. that a protection violation is impossible.

Property X is frequently stated as

X: there is an edge from vertex p to q with label α.

For these properties our protection questions do indeed look very much like transitive closure questions. Indeed if the rules R only allowed the addition of edges, then these problems would be easily solved by known methods. They are not so simple. The rules of interest to those in protection, and the particular rules we will study, allow new vertices to be added. This simple change of allowing graphs to "grow new vertices" make these problems challenging. Indeed the particular one we will study is no longer even obviously decidable.

Let us now make the above concrete by introducing the particular protection system we will study. We consider directed graphs whose arcs are labeled with an r or a w or a c. While we will manipulate these graphs as formal objects it is helpful to keep in mind the following informal semantics: A vertex corresponds to a "user", r = "read", w = "write", c = "call". If there is a directed arc from x to y with label r (respectively w,c), then x can read y (respectively write, call). We interpret this to mean that not only can x read the program and data of y but also that x can read the security information of y. (See a discussion of these issues in Section III.) For example, in the graph



x can write y, x can read z, but y cannot write z since this edge is missing. More formally, a *protection graph* is a finite, directed graph with each arc labeled by a nonempty subset of {r,w,c}. We interpret the case where an arc is labeled with other than a single element to mean that multiple "rights" are allowed.

This protection model, called the *take and grant system,* is now completed by presenting five rewriting rules.

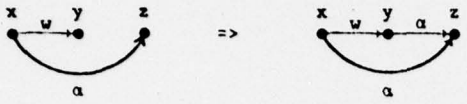
1. *Take:* Let x, y, and z be three distinct vertices in a protection graph and let there be an arc from x to y with label γ such that r ∈ γ and an arc from y to z with some label α ⊆ {r,w,c}. Then the take rule allows one to add the arc from x to z with label α yielding a new graph G'. Intuitively x *takes* the ability to do α to z from y. We will represent* this rule by

2. *Grant:* Let x, y and z be distinct vertices in a protection graph G and let there be an arc from x to y with label γ such that w ∈ γ and an arc from x to z with label α ⊆ {r,w,c}. Then the grant rule allows one to add an arc from y to z with label α yielding a new graph G'. Intuitively x *grants* y the ability to do α to z. In our representation grant is given by:

† This work was supported in part by NSF under DCR-75-07251.

†† This work was supported in part by the Office of Naval Research grant N00014-75-C-0752 and in part by NSF grant DCR74-24193.

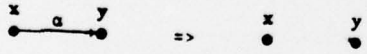
* Here and in later diagrams we abuse notation by writing an explicit right as arc label ($x \bullet\xrightarrow{r}\bullet\ y$) to mean the arc label contains that right (i.e., $x \bullet\xrightarrow{\gamma}\bullet$ such that $r \in \gamma$). We also omit the braces around sets.



3. *Create:* Let x be any vertex in a protection graph, then create allows one to add a *new* vertex N and an arc from x to N with label {r,w,c} yielding a new graph G'. Intuitively x creates a new user that it can read, write and call. In our representation

4. *Call:* Let x, y and z be distinct vertices in a protection graph G and let $\alpha \subseteq \{r,w,c\}$ be an arc from z to y and γ an arc from x to z such that $c \in \gamma$. Then the call rule allows one to add a *new* vertex N, an arc from N to y with label α, and an arc from N to z with label r yielding a new graph G'. Intuitively x is *calling* a program z and passing parameters y. The N "process" is created to effect the call: N can read the program z and can α the parameters. In our representation

5. *Remove:* Let x and y be distinct vertices in a protection graph G with an arc from x to y with label α. Then the remove rule allows one to remove the arc from x to y yielding a new graph G'. Intuitively x *removes* its rights to y. In our representation,

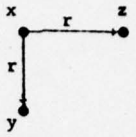


The remove rule is defined mainly for completeness, since protection systems tend to have such a rule. Moreover, we expect to study properties of protection systems other than protection violations which will use remove in a crucial way. But, for the present remove may be ignored.

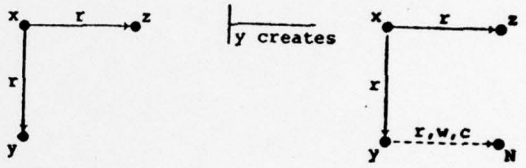
The operation of applying one of the rules to a protection graph G yielding a new protection graph G' is written $G \vdash G'$. As usual $G \vdash^* G'$ denotes the reflexive, transitive closure.

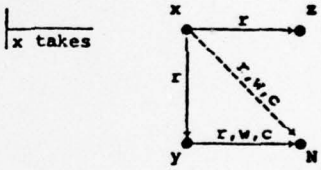
An important technical point in this system is *monotone* in the sense that if a rule can be applied, then adding arcs cannot change this. This property is crucial later. (See also [3].)

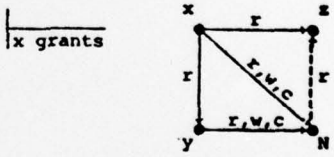
Now that we have seen the rules, let us look at their behavior. We will start with a simple question: in the graph

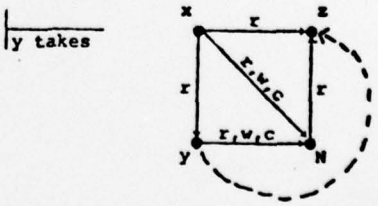


is it possible for y to r z? The answer is obviously no since there is no r arc from y to z. But we are really asking: *is there a sequence of rule applications that leads to a graph with an r arc from y to z?* More generally, say *p can* α q if there is a series of rules that leads to a graph with an arc α from p to q. Then to state our question more precisely, we ask: is it true that y can r z? Clearly, without create, the answer is no since none of the operations take, grant or call can apply. The following sequence of applications of the rules* shows that by using create the answer is yes:









* In the diagrams, dashed lines are used only as a visual aid to set off the added arcs of the current operation.

Our main theorem is stated in the next section. This theorem presents a complete answer to the question: is it true that p can α g? Indeed this theorem leads easily to a linear time algorithm for answering the question.

A final word about how this theorem contributes to our understanding of protection. Each user of a protection system needs to know:

what information of mine can be accessed by others;

what information of others can be accessed by me?

The question is vague in general, but here it is rendered in the simple question: is it true that p can α q?

The types of protection models studied here have received considerable attention recently. Our approach is related closely to the interesting work of Harrison, Ruzzo, and Ullman [3]. They show that what can be called the "uniform safety problem" is undecidable. Interpreted as a graph model, their result says that given an arbitrary set of rules (similar in spirit to take, grant, etc.) and an initial graph, it is undecidable whether or not there will ever be an arc from p to q with label α. This is a *uniform* problem in the sense that the rules are arbitrary. Even when the rules have to satisfy certain additional constraints the results of [3] and the results of Lipton and Snyder [5] show that protection is impractically complex.

Our view here is that since the uniform protection problem is so difficult and since operating systems usually require only *one* fixed set of protection rules, then the nonuniform problem should be studied. As stated before we choose the take and grant system by studying the protection literature. Note that some other nonuniform systems are trivially decidable. For example, consider a very simple system which has as its only rule, *transfer*, which is represented in our graph model as:

The transfer rule was abstracted from a survey article on security enforcement [2]. The rule says that x can give away any right it currently has. Clearly in this system p *can* α q if and only if there exists initially an x such that there is an edge from x to q with label α.

II. Basic Results

A. Subject case

Our objective is to show that there are two simple conditions that are necessary and sufficient to determine if vertex p can α vertex q. Let G be a protection graph and α ∈ {r,w,c}. Call p and q *connected* if there exists a path between p and q independent of the directionality or labels of the arcs. Define the predicates:

Condition 1: p and q are connected in G.

Condition 2: there exists a vertex x in G and an arc from x to q with label β such that

α = r implies {r,c} ∩ β ≠ ∅, or

α = w implies w ∈ β, or

α = c implies c ∈ β.

Informally, these conditions will state that p can α q if and only if there is an undirected path between p and q (condition 1) and some vertex x α's q (condition 2).

The first step is to demonstrate the necessity of conditions (1) and (2).

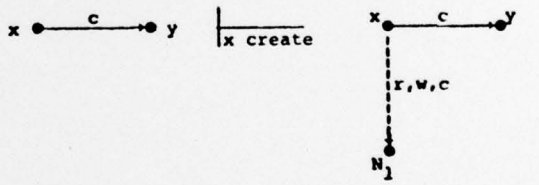
Lemma 1: Let G be a protection graph with vertices p and q and let α be a label. Then p can α q is true implies conditions (1) and (2) hold.

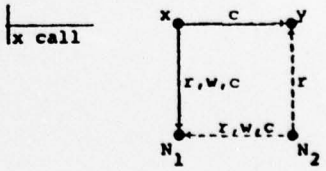
Proof: If there is an arc with label α from p to q in G then (1) and (2) are satisfied, so suppose there is no α arc from p to q in G and $G_1,\ldots,G_n$ is a sequence such that p can α q. If (1) is not satisfied in G_i then it is not satisfied in G_{i+1} since no rule application connects vertices not already connected. If (2) is not satisfied in G, let G_i be the first graph satisfying (2) and $G_{i-1} \vdash G_i$. If p is taken or granted, the choice of G_i is violated. Create cannot place an incoming arc to q, so p must be call. But regardless of what α is, p = call violates our choice of G_i. □

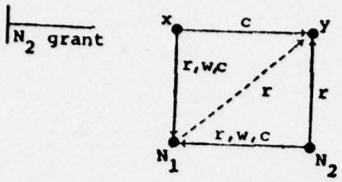
To simplify matters later and to clear up an apparent anomaly in condition (2), we next show that if a user is allowed to call another user then he is allowed to read him as well. It is this fact that allows us to write {r,c} ∩ β ≠ ∅ in condition (2) rather than just r ∈ β.

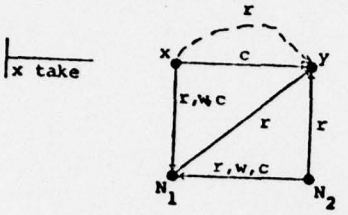
Lemma 2: In a protection graph G, x $\xrightarrow{c}$ y implies x $\xrightarrow{r,c}$ y.

Proof: Apply the following rules:









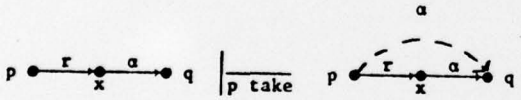
□

We next prove a key lemma that shows that the directionality and labels along a connected path are unimportant. Call vertices p and q of a protection graph *directly connected* if there is an arc between them independent of the directionality.

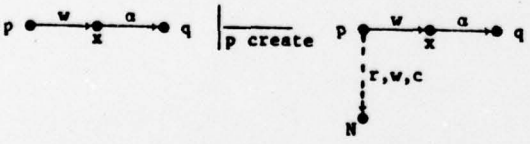
Lemma 3: Let p, q and x be distinct vertices in a protection graph, let there be an arc from x to q with label α and let p and x be directly connected. Then p can α q.

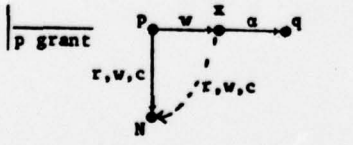
Proof: By monotonicity, there are only six distinct cases.

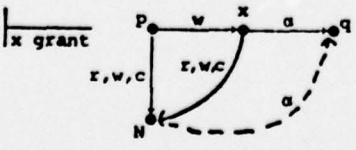
Case 1:

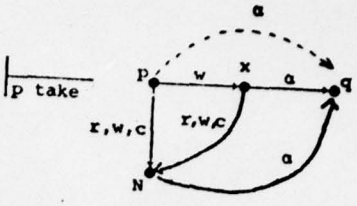


Case 2:









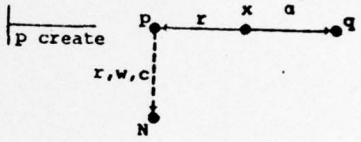
Case 3: p —c→ x —α→ q

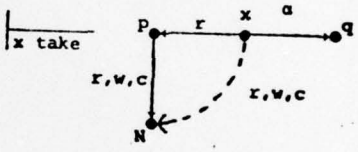
By lemma 2 this can be written as

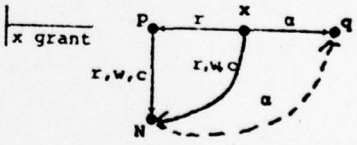
p —c,r→ x —α→ q

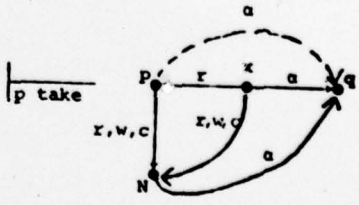
and we can appeal to case 1.

Case 4: p ←r— x —α→ q

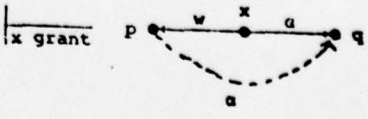






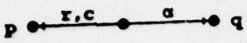


Case 5: p ←w— x —α→ q



Case 6: 

By lemma 2 this can be written as

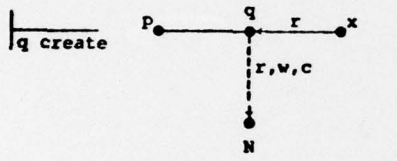


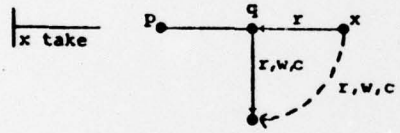
and we can apply case 4. □

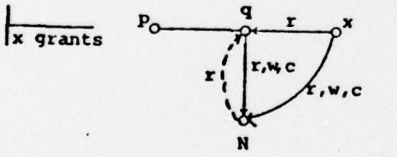
We now use lemma 3 to prove three additional lemmas to be used in the basis of our later induction.

Lemma 4: Let p, q and x be distinct vertices in a protection graph such that p is directly connected to q and there is an arc from x to q with label γ such that $\{r,c\} \cap \gamma \neq \emptyset$. Then p can r q.

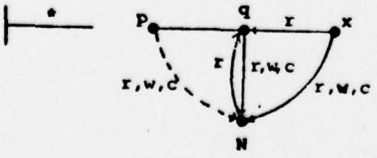
Proof: By lemma 2 we can assume that $\gamma = r$. Then we apply the following rules[+]:



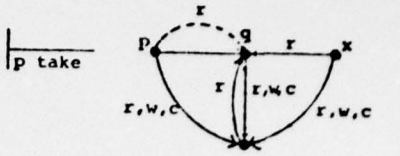




By application of lemma 3 (on the path p, q, x, N) we can realize



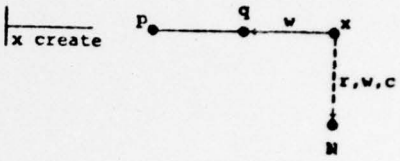
(+) p ——— q represents directly connected.

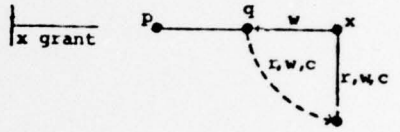


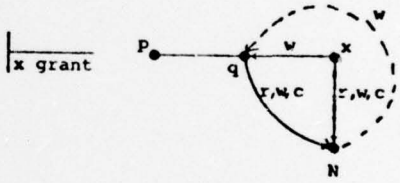
□

Lemma 5: Let p, q and x be distinct vertices in a protection graph such that p is directly connected to q and there is an arc from x to q with label γ such that $w \in \gamma$. Then p can w q.

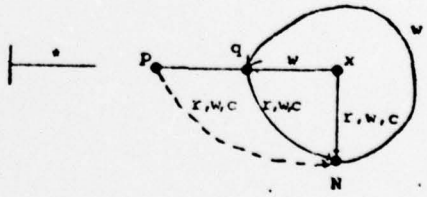
Proof: We apply the following rules:

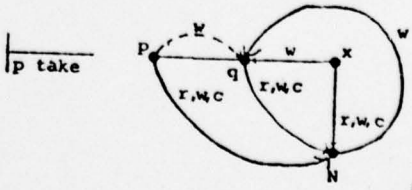






By application of lemma 3 (on path p, q, x and N) we realize

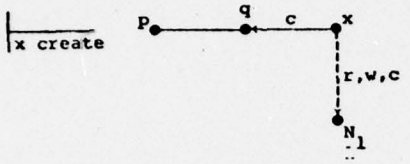


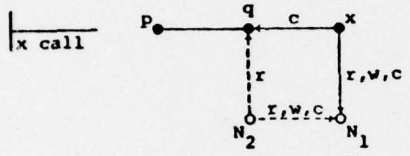


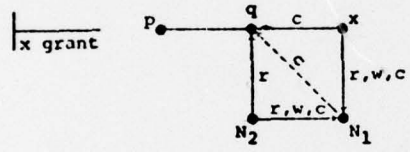
□

Lemma 6: Let p, q and x be distinct vertices in a protection graph such that p is directly connected to q and there is an arc from x to q with label γ such that $c \in \gamma$. Then p can c q.

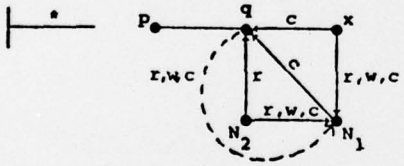
Proof: Apply the following rules:



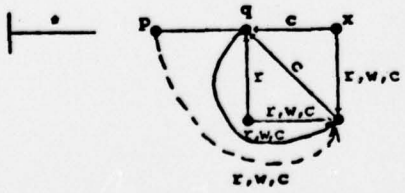




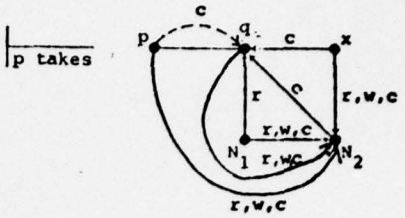
By an application of lemma 3 (on q, x, N_1) we realize



By a second application of lemma 3 (on p, q, x, N_1) we get



then



□

Theorem 1: Let p and q be distinct vertices in a protection graph and α a label. Conditions (1) and (2) are necessary and sufficient to imply p can α q.

Proof: Lemma 1 demonstrates necessity so we proceed by induction to show sufficiency. Let $p = x_n, x_{n-1}, \ldots, x_1, x_0 = q$ be the vertices on a connected path.

(Basis) For n = 1, there are two possibilities. The x guaranteed by condition (2) either coincides with $x_1 = p$ in which case the sufficiency is immediately true or else x and x_1 are distinct. By lemmas 4, 5 and 6, p can α q.

(Induction) Suppose the theorem is true for $n \geq 1$ and $p = x_{n+1}$ and x_{n+1} is directly connected to x_n. By hypothesis x_n can α a, and by lemma 3 this implies x_{n+1} can α q. □

Corollary 1: There is an algorithm for deciding if p can α q that operates in linear time in the size of the protection graph.

Proof: To verify condition (1) apply Tarjan [6]. Verifying condition (2) requires no more time than scanning the in arcs to vertex q. □

An obvious consequence of the constructions of this section is that it is simple to acquire the right to a given object if it can be acquired.

Corollary 2: If p can α q then there is an algorithm to add an arc from p to q with label α that is linear in the length of the path between p and q.

The consequence of theorem 1 is that we can precisely state the protection "policy" for our take grant system:

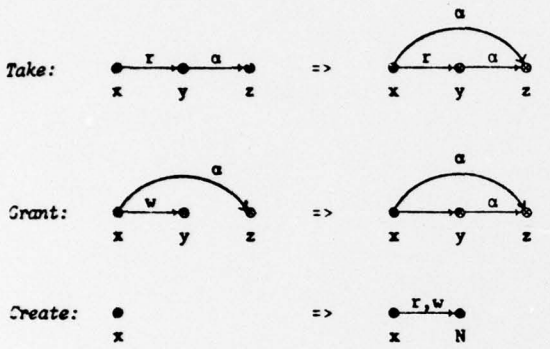
Policy: If p can initially read (write) (call) q then any user in the connected component containing p and q can also obtain the right to read (write) (read and call) q.

The policy is probably less discriminating than the reader might have expected. This is especially true considering that people usually "believe" these systems to be more discriminating. The difficulty is that up to now we have, for technical reasons, abstracted away an important distinction that is usually made for capability based security systems: the subject-object distinction.

B. Subject/Object Security

The vertices of our graphs have been thought of as "users," i.e. active agents capable of taking and granting. But these properties are not usually ascribed to files. Hence, it is customary to recognize two kinds of system components: subjects and objects. [3] In our graph model we can think of the vertices as being two colored.

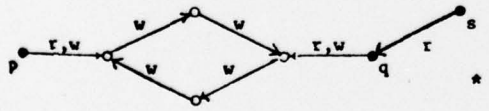
To extend our present model to incorporate objects, we define a subject-object protection graph as a finite, directed graph whose vertices are partitioned into two sets, subjects and objects, and whose arcs are labeled with {r}, {w} or {r,w}. An S-O take grant system has the following rewriting rules, where solid vertices represent subjects, open vertices represent objects and crossed vertices represent either subjects or objects.



As usual, x, y and z must be distinct.

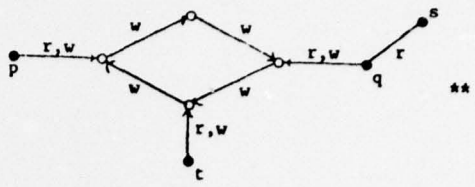
The subject-object protection graphs do not use "c", nor is the call operation defined for the S-O take grant system. We conjecture that this can be done with little difficulty, but it contributes little to the subsequent (already too complex) development.

In order to see that the above rules do in fact introduce a new set of problems, consider the following subject-object protection graph:

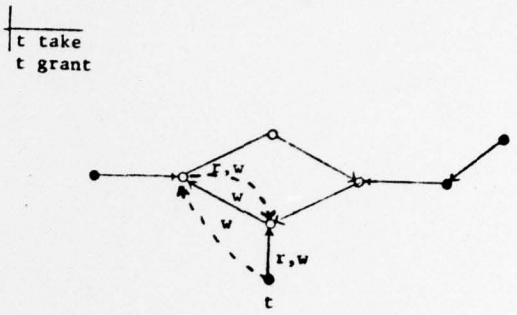


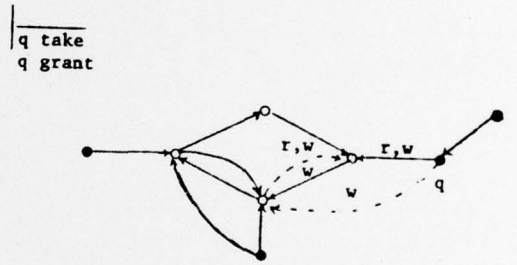
First, not that if all vertices were subjects, then by theorem 1, p can r q. As it is, p cannot r q (see theorem 2 below) even though the w's on the diamond indicate how "information" (but not "security information") could move from p to q. The reason "security information" cannot be moved around is that subjects taking and granting to accomplish its movement and objects are prevented by the rules from doing this. Accordingly, objects may be thought of as "passive" programs as well as files.

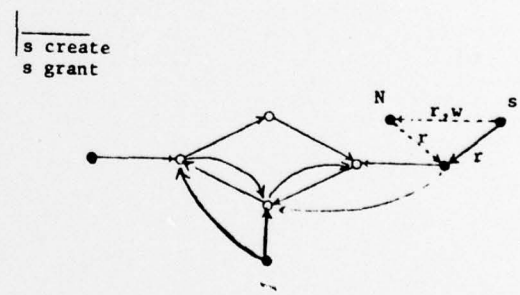
But, if we add an "agent" vertex, t, to the previous diagram,

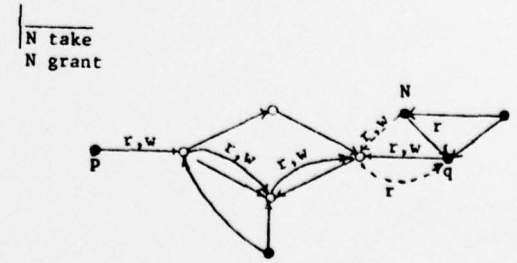


p can r q, as the following sequence establishes:









And now, by a sequence of 3 takes, p can r q.

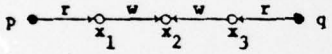
The effect of the preceding discussion is that objects can form "barriers" for security information (e.g., in diagram *) but that in closely related cases (e.g., diagram **) the barrier is ineffective. Thus, the addition of objects has increased the complexity of these systems. We dedicate the remainder of this section to establishing conditions under which p can α q for S-O take-grant systems. We only treat the "subject-subject case", i.e., when p and q are both subjects.

We will now proceed with the analysis of the SO take grant system.

Let p and q be subjects and let $x_1, \ldots, x_k$ ($k \geq 1$) be objects such that p directly connected to x_1, x_i directly connected to x_{i+1}, and x_k directly connected to q. Then we will say that p, $x_1, \ldots, x_k$, q is a *path* from p to q. With each such path we associate a *word* over the alphabet.

$$\{\vec{r}, \overleftarrow{r}, \vec{w}, \overleftarrow{w}\}$$

formed by concatenating the edge labels in the order from p to q (with the obvious interpretation: $\overset{r}{\circ\!\!\longrightarrow\!\!\circ}$ corresponds to $\vec{r}$ and so on.) For example, the path



has the word $\vec{r}\,\vec{w}\,\overleftarrow{w}\,\vec{r}$ associated with it.

Let E be the union of the following regular events:

(1) $\vec{r}(\vec{r})^+$

(2) $\overleftarrow{r}(\overleftarrow{r})^+$

(3) $(\vec{r})^*\vec{w}(\vec{r})^+$

(4) $(\overleftarrow{r})^*\overleftarrow{w}(\overleftarrow{r})^+$

(5) $(\vec{r})^+\,\vec{w}(\vec{r})^*$

(6) $(\overleftarrow{r})^+\,\overleftarrow{w}(\overleftarrow{r})^*$

where $A^+ = AA^*$. The key idea behind this definition is that paths with words that lie in E allow their subjects (i.e., end points) to "communicate". More precisely,

Lemma 7: If p, $x_1, \ldots, x_k$, q is a path with a word in E, then there is a sequence of take, grants, and creates such that p and q are directly connected.

The key to showing the decidability of the SO take and grant system is to, in a sense, obtain a converse to this lemma. In order to state this result we need several further concepts.

Let G be a subject object protection graph. Then B is a *block* of G provided B is a maximal set of subjects such that B is connected with respect to the relation: directly connected. Notice that in diagram * p and q are in different blocks while q and s are in the same block. A *bridge* between two distinct blocks is a path p, $x_1, \ldots, x_k$, q with p in one block and q in the other such that the word of the path is in E. In diagram **, there is a bridge from p's block to t's block and a brdige from t's block to the q-s block.

We are now ready to state our theorem:

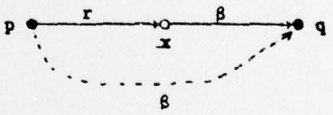
Theorem 2: Let G be a subject/object protection graph. Also let p_0, q_0 be subjects with some edge from some subject to q_0 with label $\alpha \in \{r,w\}$. Then p_0 can α q_0 if and only if

Condition 3: there exists a sequence of blocks $B_1, \ldots, B_m$ with p_0 in B_1, q_0 in B_m and for each i=1,..., m-1 there is a bridge from B_i to B_{i+1}.

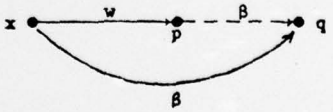
Proof: First suppose that Condition 3 is true. Then by lemma 7 there is a sequence of take and grants and creates that get p_0 and q_0 in the same block. Now by theorem 1, p_0 can α q_0. We must show that condition 3 is true. Assume that it is not.

For each path p, $x_1, \ldots, x_k$, q that joins two blocks use lemma 7 to get p, q directly connected. Let H be the resulting graph. Then in H, p can α q_0 and there are no bridges. Moreover, since condition 3 is false, p_0 and q_0 lie in different blocks. Now since p_0 can α q_0 this is a sequence of operations that will cause p_0 and q_0 to be directly connected; thus there must be a place where two vertices of different blocks are made to be directly connected. We plan to show that this is impossible.

We first observe that if two vertices of different blocks are made directly connected at some point, then there must already exist a bridge between these blocks. (But *not* necessarily a bridge in H.) In detail let p, q reside in distinct blocks in H' ($H \mid\!\overset{-}{*}\, H'$) and some operation add an edge from p to q with label β. Then if this operation is a take

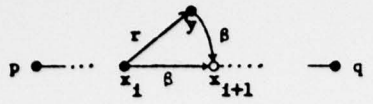


it follows that p, x, q is a bridge between the blocks. On the other hand, if this operation is a grant

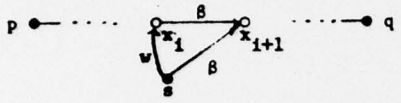


Then p, q are already in the same block which is impossible Thus we know that there is a sequence of operations on H that creates a bridge between two distinct blocks. Let H' be such that there is no such bridge in H' and H" has one where $H \mid\!\overset{-}{*}\, H' \mid\!- H''$.

Now we need only argue that H' must already have a brdige to complete the proof. Let p, $x_1, \ldots, x_k$, q be the bridge in H". Clearly one of its edges was added by either a take or grant from H'. First assume that it was a take. Then (convention: $p = x_0$, $q = x_{k+1}$)



for some vertex y and some label β. (We have assumed the edge goes from x_i to x_{i+1}; the dual case is similar.) By the definition of path, x_i must equal p, i.e., i = 0. If y is a subject then y, x_1,..., x_k, q is already a bridge; if y is an object then p, y, x_1,..., x_k, q is already a bridge since if $\vec{\beta}\delta$ is in E then so is $\vec{r}\vec{\beta}\delta$. Thus the operation was not a take. It can then only be a grant. Thus,



for some vertex s and label β. (We have again assumed a direction without loss of generality.) Now s is in the same block as q. For either x_{i+1} = q or s, x_{i+1},..., x_k, q is a bridge: the latter uses the fact that E is closed under suffix. Thus $x_i \neq p$. Now we claim that p, x_1,..., x_i, s is a bridge which is impossible: If β = r, then we are using the fact that

$$\delta\vec{r}\lambda \text{ in E implies } \delta\vec{w} \text{ in E.}$$

If β = w, then we are using the fact that

$$\delta\vec{w}\lambda \text{ in E implies } \delta\vec{w} \text{ in E.}$$

Therefore we have reached a contradiction and the theorem is proved. □

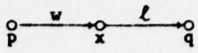
Evidently, the S-O take and grant system provides for a more discriminating policy than the take and grant system.

Corollary 3: For the SO take and grant system there is an algorithm to decide if p can α q.

III. Discussions

We have used a "generalization" of transitive closure in order to abstract the behavior of two kinds of protections systems: those with just subject components and those with subject-object components. Our choice of primitive rules has been strongly motivated by the protection literature. But we do not believe we have defined the only interesting set of protection rules. There are probably many others and we expect that much work remains in identifying sets that may be efficiently verified as well as having highly discriminating policies.

Another direction for research is to add *inert* rights. For example consider a protection graph G where there is an edge from vertex x to vertex q with label ℓ where ℓ is a new type of label. We then wish to know if p can ℓ q, i.e. if there is a series of takes, grants, creates, and calls that lead to a graph with an edge from p to q with label ℓ. The key to this problem is that while label ℓ can be taken and granted it has no special role(s) as r,w, and c do. The label ℓ is simply something that is passed around, and that is all. A graph such as



shows that our theorem 1 is no longer true.

Another way to modify our system is to control the amount of cooperation necessary to obtain a particular right. With each rule application the vertex that is denoted x in our definitions will be called a *conspirator*. Thus in

x is a conspirator. Then an interesting question is can p *can* α q with at most m conspirators.

One might then hope to attach some kinds of likelihoods in a precise way to whether or not a system is secure.

In general there are many other problems to be studied. All of these problems are in a sense generalizations of transitive closure. The key and most important aspect of this generalization is that the most interesting rules allow "growth", i.e. the addition of new vertices. It appears that understanding the structure of such problems is interesting beyond its application to the study of protection models.

References

1. E. Cohen.
   Ph.D. Thesis (in progress), Carnegie-Mellon University, 1976.

2. G. S. Graham and P. J. Denning.
   Protection principles and practice.
   AFIPS Conference Proceedings 40:417-429, 1972.

3. M. A. Harrison, W. L. Ruzzo, and J. D. Ullman.
   On protection in operating systems.
   Proceedings of the 5th annual SIGOPS Conference, 1975.

4. A. K. Jones.
   Protection in programmed systems.
   Ph.D. Thesis, Carnegie-Mellon University, 1973.

5. R. J. Lipton and L. Snyder.
   Synchronization and security.
   In preparation, 1976.

6. R. E. Tarjan.
   Depth first search and linear graph algorithms.
   SIAM J. Computing 1:146-160, 1972.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) RR-103	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER (9) Research rept.
4. TITLE (and Subtitle) (6) A Linear Time Algorithm for Deciding Security	5. TYPE OF REPORT & PERIOD COVERED: Technical
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Anita K. Jones, Richard J. Lipton, Lawrence Snyder	8. CONTRACT OR GRANT NUMBER(s) (15) N00014-75-C-0752, NSF DCR-74-24193
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Yale University, Computer Science Department, 10 Hillhouse Ave, New Haven, CT 06520	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Information Systems Program, Arlington, Virginia 22217	12. REPORT DATE (11) OCT 1976; 13. NUMBER OF PAGES (12) 12 pp.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified; 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

407 051

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

prtection
security
take and grant
subject
object
linear time
transitive closure
operating system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The take-grant security system is introduced. Active security agents (subjects) and passive entities (objects) are recognized. A linear time algorithm for recognizing security violations follows from three conditions presented that are proved to be necessary and sufficient for system security.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)